МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ИНТЕГРАЦИЯ ЭЛЕКТРОННОЙ БИБЛИОТЕКИ ИРБИС64+ В СИСТЕМУ ДИСТАНЦИОННОГО ОБУЧЕНИЯ IPSILONUNI

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Андриянова Дениса Александровича

Научный руководитель
зав. каф. к.ф.-м.н. доцент _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2021

ВВЕДЕНИЕ

Актуальность темы. Цифровизация образовательных процессов на данный момент — одно из самых приоритетных направлений в России. Так, например, с 1 сентября 2020 года, в соответствии с проектом постановления правительства РФ, в ряде регионов России начнется глобальный эксперимент по внедрению «цифровой образовательной среды» в школах и колледжах. Об этом сообщило Министерство просвещения России.

Современная образовательная система обязует преподавателей составлять множество отчётов и документов. Одним из таких документов является рабочая программа. Разработка рабочей программы, представляющая собой достаточно сложный учебный и нормативный документ, требует от автора-составителя высокого уровня квалификации, а также больших затрат по времени. Информационные технологии позволяют серьёзно упростить такие задачи, предоставляя удобные инструменты для создания и редактирования таких документов.

Цель бакалаврской работы — рассмотрение функционала электронной библиотеки СГУ и последующей её интеграции в систему дистанционного обучения IpsilonUni.

Поставленная цель определила **следующие задачи**: изучение спецификаций ИРБИС64+ и IpsilonUni, реализация загрузки данных о литературных источниках из базы данных ИРБИС, парсинг данных и преобразование в удобный формат, разработка Веб-API для пересылки данных на IpsilonUni.

Практическая значимость бакалаврской работы заключается в том, что интеграция электронной библиотеки в IpsilonUni позволило бы преподавателям удобно и просто ссылаться на источники литературы при составлении рабочих программ.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников, трёх приложений. Общий объём работы — 72 страницы, из них 40 страниц — основное содержание, включая 20 рисунков, листинг кода в качестве приложения, список использованных источников информации — 24 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теоретическая часть» посвящен основным техноло-

гиям, использующимся при реализации Веб-API и загрузки данных о литературных источниках из базы данных ИРБИС.

В ходе подготовки к интеграции в первую очередь необходимо разобраться в системе IpsilonUni, а также в технологиях и принципах построения ПО, на базе которых она функционирует, а именно фреймворк Ruby on Rails и паттерн проектирования MVC. Это было необходимо, так как нужно было учитывать специфику этой системы и подобрать наиболее удобное решение для последующей интеграции. В этом шаге имелось множество проблем, основной из которых была несовместимость технологий, на которых были разработаны IpsilonUni и ИРБИС64+. Отсюда было принято решение реализовать «посредник», а именно API, который позволил бы доставлять информацию из базы данных ИРБИС на IpsilonUni в удобном и понятном для него формате.

Для реализации API необходимо выбрать технологию, которая бы помогла создать гибкую и расширяемую систему, а также выделить основные принципы при его построении. В результате был выбран фреймворк ASP.NET WebApi, который предоставляет удобные инструменты для создания Веб-API. Это позволило бы запрашивать всю необходимую информацию с системы IpsilonUni при помощи удобного протокола HTTP в формате JSON.

Так как этот программный интерфейс будет взаимодействовать с базой данных ИРБИС, необходимо подробно изучить её спецификацию, внутреннюю структуру и формат хранения данных, а также разработать функционал для подключения к этой базе. В этом шаге имелось множество проблем, так как либо часть документации отсутствовала, либо имела очень сложную для понимания структуру.

Также для предстоящего локального тестирования и развертки IpsilonUni и полученного WebApi, необходимо изучить программное обеспечение Docker для контейнеризации приложений, а также инструменты виртуализации.

В подразделе 1.1 рассматриваются принципы построения API (Application Programming Interface). API — это вычислительный интерфейс, который определяет взаимодействие между несколькими программными посредниками. Он определяет типы вызовов или запросов, которые могут быть сделаны, как их делать, форматы данных, которые следует использовать, соглашения, которым нужно следовать, и т.д. Он также может предоставлять механизмы расширения, чтобы пользователи могли расширять существующие функциональные

возможности различными способами и в разной степени. API может быть полностью настраиваемым, специфичным для компонента или спроектированным на основе отраслевого стандарта для обеспечения взаимодействия. Благодаря скрытию информации API-интерфейсы обеспечивают модульное программирование, позволяя пользователям использовать интерфейс независимо от реализации.

В подразделе 1.2 подробно описывается технология ASP.NET WebApi, которая представляет способ построения приложения ASP.NET, который специально заточен для работы в стиле REST (Representation State Transfer или «передача состояния представления»). REST-архитектура предполагает применение следующих методов или типов запросов HTTP для взаимодействия с сервером:

- GET;
- POST;
- PUT;
- DELETE.

Зачастую REST-стиль особенно удобен при создании всякого рода Single Page Application, которые нередко используют специальные фреймворки типа Angular, React или Vue.js. По сути Web API представляет собой веб-службу, к которой могут обращаться другие приложения. Причем эти приложения могут представлять любую технологию и платформу — это могут быть приложения под веб, мобильные или десктопные клиенты.

Средство Web API основано на добавлении в приложение ASP.NET MVC Framework контроллера специального вида. Эта разновидность контроллеров, которая называется контроллером API, обладает двумя характеристиками:

- методы действий возвращают объекты моделей, а не такие объекты, как например типа ActionResult;
- методы действий выбираются на основе HTTP-метода, используемого в запросе.

Объекты моделей, возвращаемые методом действия контроллера API, кодируются в формате JSON и отправляются клиенту. Контроллеры предназначены для доставки веб-служб данных, поэтому они не поддерживают представления, компоновки или любые другие средства, которые применялись для генерации HTML-разметки в примере созданного приложения.

Отсутствие возможности у контроллера API генерации разметки HTML из представлений является причиной, по которой в одностраничных приложениях комбинируются стандартные приемы ASP.NET MVC с Web API. Инфраструктура ASP.NET MVC Framework выполняет шаги, требуемые для доставки HTML-содержимого пользователю (включая аутентификацию, авторизацию, выбор и визуализацию представления). После того, как HTML-содержимое доставлено в браузер, запросы Ajax, генерируемые содержащимся внутри кодом JavaScript, будут обрабатываться контроллером Web API.

Как демонстрировалось ранее, в обычных контроллерах можно создавать методы действий, которые возвращают данные JSON для поддержки Ajax, но контроллер API предлагает альтернативный подход. Этот подход предусматривает отделение действий, относящихся к данным, от действий, связанных с представлением, и делает создание универсальных приложений Web API быстрым и простым.

В подразделе 1.3 подробно описывается система ИРБИС64+, которая представляет собой систему автоматизации, предназначенную для создания и ведения электронной библиотеки (в отличие от ИРБИС64, который предназначен для ведения электронного каталога).

Пользователи под «электронной библиотекой» понимают самые разные вещи. Некоторые называют электронной библиотекой электронный каталог, где к библиографическим описаниям привязаны ссылки на полные тексты, которые не участвуют в поиске и выступают в роли ресурса для просмотра, — такую «электронную библиотеку» можно создавать с помощью обычного ИРБИС64. Другие пользователи называют электронной библиотекой то, что можно создавать с помощью другого продукта – ИРБИС64 ПБД, — т.е. полнотекстовые базы данных, в которых можно осуществлять поиск по словам. ИРБИС64+ — нечто более широкое. Прежде всего важно отметить, что основное назначение электронной библиотеки с точки зрения читателя — это поиск нужной информации, в то время как назначение электронного каталога — поиск нужной книги (издания).

В общем функционал, обусловленный требованиями именно электронной библиотеки, состоит из следующих элементов:

— полнотекстовый поиск на основе автоматического разбиения текстов на страницы и их пословной индексации;

— представление результатов поиска в порядке убывания релевантности документов, а в рамках одного документа – в порядке убывания релевантности найденных страниц;
— предоставление конечному пользователю возможности просмотра полных текстов постранично с обеспечением навигации по релевантным страницам и маркировки найденных слов запроса;
— развитая система прав доступа конечных пользователей к полным текстам;
— система учета обращений пользователей к полным текстам.

Таким образом, основой ИРБИС64+ являются базы данных, представляющие собой совокупность связанных библиографических данных и полных текстов изданий.

Библиографические данные ведутся на основе коммуникативного формата RUSMARC. В качестве полных текстов используются распознанные PDF-файлы, которые подвергаются автоматическому разбиению на страницы и индексированию по словам.

Поиск в системе ведется по словам полного текста и/или любым элементам библиографического описания. Результат поиска в общем случае представляется в виде списка документов (т.е. библиографических описаний изданий) в порядке убывания их релевантности. При этом с каждым документом связан список релевантных страниц полного текста (также в порядке ее убывания). Релевантность определяется на основе оригинального критерия, который учитывает количество и контекстную близость слов запроса, найденных в полном тексте. При сравнении слов используется механизм морфологии русского языка. Найденные термины на страницах полного текста маркируются цветом.

В подразделе 1.4 описывается паттерн проектирования веб-приложений MVC, который включает в себя несколько более мелких шаблонов. При использовании MVC на три отдельных компонента разделены модель данных приложения, пользовательский интерфейс и логика взаимодействия пользователя с системой, благодаря чему модификация одного из этих компонентов оказывает минимальное воздействие на остальные или не оказывает его вовсе.

Основная цель применения MVC состоит в разделении данных и бизнес-логики от визуализации. За счет такого разделения повышается возможность повторного использования программного кода: например, добавить представ-

ление данных какого-либо существующего маршрута не только в виде HTML, но и в форматах JSON, XML, PDF, XLSX становится очень просто и не требует исменений слоя бизнес-логики исходного маршрута. Также упрощается и сопровождение программного кода: внесение изменений во внешний вид, например, не отражаются на бизнес-логике, а изменения бизнес-логики не затрагивают визуализацию.

В подразделе 1.5 рассматривается MVC-фреймворк для построения веб-приложений Ruby on Rails. Фреймворк написан на языке программирования Ruby. Rails подходит как для разработки обычных сайтов, которые должны быть реально быстрыми, отказоустойчивыми и работающими под высокой нагрузкой, так и для веб-приложений со сложной бизнес-логикой и динамичными web-интерфейсами. Ruby on Rails является открытым программным обеспечением и распространяется под лицензией MIT.

В подразделе 1.6 описывается платформа для разработки, доставки и запуска контейнерных приложений Docker и его использование вместе с Rails. Docker — это платформа для разработки, доставки и запуска контейнерных приложений. Docker позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом. Он позволяет запускать множество контейнеров на одной хост-машине.

Контейнеризация похоже на виртуализацию, но это не одно и то же. Виртуализация работает как отдельный компьютер, со своим виртуальным оборудованием и операционной системой. При этом внутри одной ОС можно запустить другую ОС. В случае контейнеризации виртуальная среда запускается прямо из ядра основной операционной системы и не виртуализирует оборудование. Это означает, что контейнер может работать только в той же ОС, что и основная. При этом так как контейнеры не виртуализируют оборудование, они потребляют намного меньше ресурсов.

Docker используется вместе с Rails, так как он обеспечивает, при использовании приложения большой аудиторией. Docker дает большие преимущества масштабируемости за счет динамического развертывания контейнеров. Это означает, что приложение может увеличивать или уменьшать количество экземпляров в режиме реального времени.

Возможно использовать контейнеры с виртуальными машинами для увеличения плотности приложений с помощью микросервисной архитектуры. На-

пример, вместо запуска трех служб на трех отдельных виртуальных машинах можно запускать три службы в трех контейнерах на одной виртуальной машине. Меньшее количество виртуальных машин означает, что обслуживание инфраструктуры будет не так дорого.

Также это отличное сочетание инструментов для команд. Без Docker было бы много работы по стандартизации операционной системы каждого разработчика и версий каждого инструмента. Использование Docker обеспечивает то, что среда приложения находится внутри контейнера и автоматически обеспечивает стандартизованную среду для каждого разработчика.

Наконец, Docker идеально подходит для приложений Rails, которые имеют множество зависимостей и других необходимых технологий. Подключение к внешним технологиям, таким как PostgreSQL или MS SQL, может быть проблемой в Rails.

Docker исправляет это, позволяя установить зависимости и соединения в одном файле и упаковать его для загрузки как одного.

В подразделе 1.7 описывается понятие виртуализации и ORACLE VM VIRTUALBOX. Виртуализация — это процесс создания программного (или виртуального) представления чего-либо, например виртуальных приложений, серверов, хранилищ и сетей. Это единственный и самый эффективный способ сокращения расходов на ИТ-инфраструктуру при одновременном повышении эффективности и адаптивности для компаний любых размеров.

VirtualBox позволяет пользователям создавать, импортировать и запускать на своих компьютерах несколько виртуальных машин одновременно, будь то старые и с неодинаковыми версиями Windows или с другими операционными системами, такими как Linux или Mac OS X. В программе предусмотрены установки по умолчанию для новеньких пользователей и при этом имеются богатые возможности настройки для более опытных, например, тонкая настройка выделенной памяти и размера жесткого диска.

Второй раздел «Реализация функционала для интеграции электронной библиотеки» посвящён шагам и трудностям, с которыми пришлось столкнуться при интеграции электронной библиотеки и написания API, который в последствии будет выступать прослойкой между электронной библиотекой и серверной частью системы IpsilonUni.

В подразделе 2.1 описывается процесс изучения функционала электрон-

ной библиотеки. Чтобы получать данные о литературе на IpsilonUni необходимо было разобраться, как и в каком виде они хранятся в ИРБИС64+ и каким образом их можно передавать туда. Это стало одним из самых сложных этапов настоящей работы.

ИРБИС64+ — довольно старое ПО с очень специфичной структурой и сложными для понимания модулями. Готовые модули ИРБИС64+, а именно J-ИРБИС и WEB-ИРБИС, либо не предоставляют открытого API, либо имеют слишком сложную для понимания спецификацию. Основная информация по всем модулям получалась на форумах из обсуждений пользователей, а также у сотрудников библиотеки, что в рамках текущей задачи оказалось слишком трудозатратным.

Ещё одной проблемой стало то, что IpsilonUni запускается на системах с операционной системой Linux, а ИРБИС работает только под операционными системами WINDOWS, в следствии чего, при разработке, тестирование запросов напрямую к базам данных электронной библиотеки не представляется возможным.

В открытых источниках, среди готовых решений, была найдена библиотека ManagedIrbis, написанная под язык C#, но у неё отсуствовала какая-либо документация (информацию по библиотеки можно было найти только на форуме ИРБИС в ответах самого разработчика) и поддержка разработчика (последнее обновление библиотеки производилось в 2014 году). Однако её использование было наиболее понятным и простым, так как в ней имелись несколько примеров проектов. С её помощью удалось подключиться и получить хоть какие-то тестовые данные с ИРБИС. Было принято решение написать Веб-API на языке C#, которое принимало бы HTTP-запросы с IpsilonUni, получало и парсило данные с ИРБИС64+ с помощью библиотеки ManagedIrbis, переводило их в более понятный формат (в данном случае идеальным вариантом был JSON) и возвращало их в качестве ответа.

Подраздел 2.2 посвящён разбору спецификации ИРБИС, а также разработке классов для подключения к базам данных. В справке к ИРБИС64+ были найдены соответствующие тэги к свойствам, которые были указаны выше, и был составлен соответствующий список:

— MFN — номер источника в базе данных ИРБИС;
— основное заглавие — Поле 200, подполе A;

— список авторов — Поле 205;
— издательство — Поле 210, подполе C;
— год публикации — Поле 210, подполе D;
— город публикации — Поле 210, подполе A;
— индекс УДК — Поле 675;
— является ли источник литературы онлайн-ресурсом — Поле 951;
— ссылка на источник — Поле 951, подполе I;
— количество экземпляров — Поле 910;
— количество свободных экземпляров — Поле 910.

Для подключения к базам данных ИРБИС64+ используется класс IrbisConnection, который в последствии необходим для разработки веб-API. Открытие соединения с базой данных происходит в три этапа:

1. создаётся экземпляр класса IrbisConnection;
2. вызывается метод ParseConnectionString, в которую, в качестве аргумента, передаётся строка подключения, где описывается информация, необходимая для открытия соединения;
3. вызывается метод Connect.

В конструкторе класса производится начальная инициализация соединения. Затем в методе ParseConnectionString входная строка сначала разбивается на свои составляющие. Затем из составляющих извлекается необходимая информация, такая как адрес сервера, логин, пароль и т.д. Если строка имеет неправильный формат, то выбрасывается исключение IrbisException. В методе Connect происходит непосредственное подключение к базе данных по протоколу TCP/IP.

После успешного подключения можно использовать метод Search, который возвращает номера записей, удовлетворяющих аргументу, являющимся поисковым запросом для баз данных ИРБИС. Для того, чтобы прочитать саму запись, необходимо вызвать метод ReadRecord, аргументом которой является номер записи в базе.

В подразделе 2.3 описывается процесс разработки веб-API. , которое позволило бы по HTTP-запросу подключаться к серверу ИРБИС64+, производить там поиск и возвращать необходимые данные в удобном для IpsilonUni формате — JSON.

В веб-API ASP.NET контроллер — это класс, обрабатывающий HTTP-

запросы. Открытые методы контроллера называются методами действий или просто действиями. Когда платформа веб-API получает запрос, она направляет запрос в действие. Чтобы определить, какое действие следует вызвать, платформа использует таблицу маршрутизации. В статическом классе WebApiConfig создаётся основной маршрут. Всю логику создания ответа на HTTP-запрос содержит класс под названием BooksController. Обработка Get запроса по адресу api/Books производит одноименный метод.

Метод принимает три необязательных входных параметра:

— query - строка для поиска;
— searchType - тип поиска (по автору или заглавию);
— count - количество искомых записей.

С помощью раннее описанного класса IrbisConnection, открывается активное соединение с ИРБИС.

Для удобства и возможности быстрого редактирования, строка подключения задаётся в конфигурационном файле Web.config.

После открытия соединения формируется запрос к базе данных ИРБИС. Метод SearchTypeToIrbisSearchType преобразует входной параметр контроллера searchType, который задаётся в HTTP-запросе, в формат запроса ИРБИС. Формат запроса имеет вид $X = QUERY$, где X — поле, по которому будет производиться поиск, а $QUERY$ — строка для поиска. Затем производится поиск и отбирается необходимая информация и возвращается ответ в формате JSON.

В подразделе 2.4 описывается процесс развёртки полученного приложения ASP.NET WebApi на локальном сервере IIS. Традиционно веб-сервер IIS (Internet Information Services) применялся для развертывания веб-приложений. Для хостирования веб-приложений ASP.NET Core также может применяться IIS, только в отличие от предыдущих версий ASP.NET теперь его роль будет сводиться к прокси-серверу. Хостирование приложений ASP.NET Core на IIS происходит с помощью нативного модуля AspNetCoreModule, который сконфигурирован таким образом, чтобы перенаправлять запросы на веб-сервер Kestrel. Этот модуль управляет запуском внешнего процесса dotnet.exe, в рамках которого хостируется приложение, и перенаправляет все запросы от IIS к этому хостирующему процессу.

В результате действий, приведенных в этом разделе приложение WebApi

будет развёрнуто и появится возможность обращаться к нему по указанному в настройках адресу (в примере localhost).

ЗАКЛЮЧЕНИЕ

В настоящей работе были рассмотрены современные технологии для создания веб-API, такие как ASP.NET WebApi, система автоматизации, предназначенная для создания и ведения электронной библиотеки ИРБИС64+, и портал дистанционного обучения IpsilonUni, а также было разработанно приложение Веб-API для последующий интеграции библиотеки на портал IpsilonUni. Это API позволяет очень удобно получать информацию об источниках литературы с баз данных электронной библиотеки с помощью HTTP-запроса. В дальнейшем разработанный функционал будет выступать прослойкой между системой IpsilonUni и ИРБИС64+ и с его помощью преподаватели смогут намного проще и быстрее добавлять ссылки в раздел источников своих рабочих программ, а также гарантировать корректность этого раздела.

Все материалы и код приложения выложены на репозиторий Git, который можно просмотреть по ссылке https://github.com/NyOxCF/SSU.Library/tree/master.